Case Study: Cyclistic Bike-Share Analysis

1. A clear statement of the business task

The company Cyclistic, a bike-share company in Chicago, wants to maximize the number of annual memberships, as annual members are more profitable than casual riders. With this objective, the marketing team needs to understand how annual members and casual riders use Cyclistic bikes differently.

The task is to perform a data-driven analysis of 12 months of historical trip data to identify usage patterns, behavioral differences, and trends between the two rider groups.

These insights will guide the design of targeted marketing strategies aimed at converting casual riders into annual members, because the company will be able to know which aspects should the marketing focus on to increase the chances of a casual rider becoming an annual member.

2. A description of all data sources used

This analysis uses 12 months of Cyclistic bike-share trip data, obtained from here, covering the period January 2025 to December 2025 (files 202501-divvy-tripdata.csv through 202512-divvy-tripdata.csv) (the datasets have a different name because Cyclistic is a fictional company).

Each month is provided as a separate CSV file, using points for decimals numbers (standard numeric formatting), where each row represents a single bike trip. All files share a consistent schema with 13 columns, including

1. "ride_id": string, combination of 16 characters of numbers and upper letters.
2. "rideable_type": string, can have the value classic_bike or electric_bike.
3. "started_at": datetime, start datetime of trip.
4. "ended_at": datetime, end datetime of trip.
5. "start_station_name": string, name of the station where the trip started.
6. "start_station_id": string, id of starting station.
7. "end_station_name": string, name of the station where the trip ended.
8. "end_station_id": string, id of ending station.
9. "start_lat": float, latitude of trip start.
10. "start_lng": float, longitude of trip start.
11. "end_lat": float, latitude of trip end.
12. "end_lng": float, longitude of trip end.
13. "member_casual": string, can have the value member or casual.

The member_casual field is essential for answering the business question, as it allows the comparison of annual members and casual riders.

The data is first-party operational data provided by Motivate International Inc., the official operator of Chicago's bike-share system. It covers a full year and includes all trips, not only a sample. Data is released monthly and reflects real trip records, which can be found here.

The data has been made available by Motivate International Inc. under this license. It excludes personally identifiable information, which ensures compliance with data privacy and security standards and it is accessible without restrictions, always following the license.

Data Integrity Verification

To verify data integrity, we analysed the file 202501-divvy-tripdata.csv in Excel and verified:

- Valid timestamps (ended_at > started_at)
- Missing station names or IDs (present in some rows)
- Uniqueness of ride_id by removing any duplicates
- Some rides start and end at the same station (may be legitimate or errors)

We confirmed that the data is relevant to our business question ("How do annual members and casual riders use Cyclistic bikes differently?") because the dataset contains the necessary fields to analyze rider behavior across time, location, and bike type. From it, we

can extract weekday patterns, seasonal trends, ride lengths, peak usage times, and station popularity, all of which help determine how casual riders and members differ.

3. Documentation of data cleaning and manipulation

Tools used and rationale

We used Excel + Power Query for initial cleaning (Annex 1) because:

- It handles CSV files efficiently
- Power Query automates the process across all 12 files
- All cleaning steps are saved in the “Applied Steps” panel
- Queries can be refreshed automatically
- This ensures reproducibility and reduces manual error

Ensuring data integrity

The original CSV files (202501–202512) were stored in a separate folder, and all transformations were carried out on copies.
For the first cleaning of data, we only worked with 202501-divvy-tripdata.csv in a sheet of an Excel file. Once we confirmed the data was cleaned, we added the other CSV files to different sheets of the same Excel file and the same Power Query steps were applied.

Cleaning steps

I. Import and preparation

- Loaded each CSV into Excel using Power Query, ensuring consistent parsing.
- Set regional settings to English (United States) to avoid decimal separator and comma-delimiter conflicts.

II. Data validation and filtering

- Removed duplicate rows to confirm ride_id uniqueness.
- Filtered out rows where start_station_name or end_station_name were null, which also removed corresponding null station IDs.
- Ensured valid timestamps by creating a ride_time_length column (ended_at – started_at) and filtering for positive durations only.

III. Feature engineering

- ride_time_length: Duration of each trip, used for comparison between rider groups.
- day_of_week: Extracted using

- =Date.DayOfWeek(Date.From([started_at])) + 1 (Sunday = 1, , Monday = 2, etc.).
- distance_stations: Estimated distance between start and end points using a flat-Earth approximation (considers that 111 km ≈ 1° latitude and similar for longitude, just adjusted by cos(latitude)). This metric is accurate within 1% for short urban trips and useful for comparing spatial behavior. The formula was =Number.Sqrt(Number.Power(([end_lat]-[start_lat])*111,2) + Number.Power(([end_lng]-[start_lng])* 111 * Number.Cos(([end_lat] + [start_lat]) / 2 * Number.PI / 180),2)).

Verification of clean data

- Confirmed all remaining rows had valid station names and IDs
- Checked statistical summaries of ride_time_length
- Verified uniqueness of ride_id
- Performed random manual spot-checks

4. Summary of the analysis

Data organization for analysis

After cleaning and preprocessing the dataset in Excel and Power Query, all monthly files were combined into a single dataset in a CSV file and exported to a highly efficient Parquet file. This format enabled fast loading and consistent data types in Python (Annex 2).
Once loaded into pandas, additional fields were created to support deeper analysis, such as month (month of each ride) and day (day of month).

We also had to recalculate ride_time_length (ride duration in seconds), because it did not load correctly, and we changed day_of_week from numbers to Sunday–Saturday This structured dataset allowed effective aggregation, filtering, and statistical computations.

Data formatting and preparation

The data was properly formatted before analysis:

- started_at and ended_at were converted to datetime.
- ride_time_length was recomputed in seconds to ensure accuracy.
- Demographic-like fields (member_casual, rideable_type) were kept as categorical, improving memory efficiency.
- Outliers such as negative ride durations were removed.
- day_of_week was mapped to ordered categories (Sunday → Saturday) to ensure correct chronological grouping.

Surprises discovered in the data

During analysis, several unexpected patterns emerged:

- Casual riders had significantly longer ride durations, often more than double the average of annual members.
- Members took far more rides overall, especially on weekdays, but their rides were usually shorter and more consistent in length.
- A noticeable portion of trips had identical start and end coordinates, suggesting short loops, which may reflect leisure behavior or small GPS drift.
- Weekend usage spiked dramatically for casual riders.
- Despite expectations, members also had non-trivial weekend usage, though still dominated by weekday commuting patterns.
- The share of classic bikes rises notably in warmer months (e.g., June: casual 56.6%, members 56.8%) and drops in winter (e.g., December: casual, 36.9%, members 48.9%), indicating a seasonal shift toward electric bikes in colder months.

Key trends and relationships

1. Ride Duration

- Casual riders consistently recorded longer rides.
- Members had shorter, more predictable durations, supporting commuting or short-errand behavior.

2. Distance Between Station

- Casual users usually covered longer distances on average than members, but the difference is not much.

3. Day-of-Week Patterns

- Casual ridership peaked on weekends, especially Saturday.
- Members had the opposite pattern: highest ridership on weekdays, with a drop on weekends.
- Average ride time for casual riders increased dramatically on weekends, while members remained stable.

4. Monthly/Seasonal Trends

- Both groups saw strong seasonal patterns, with higher usage in warm months.
- Casual ridership rose more sharply in summer, further supporting leisure behavior.
- Member usage stayed relatively more stable across months, but there is also a preference for warmer months.
- Usage of classical bikes was reduced in cold weather for both types.

How these insights answer the business question

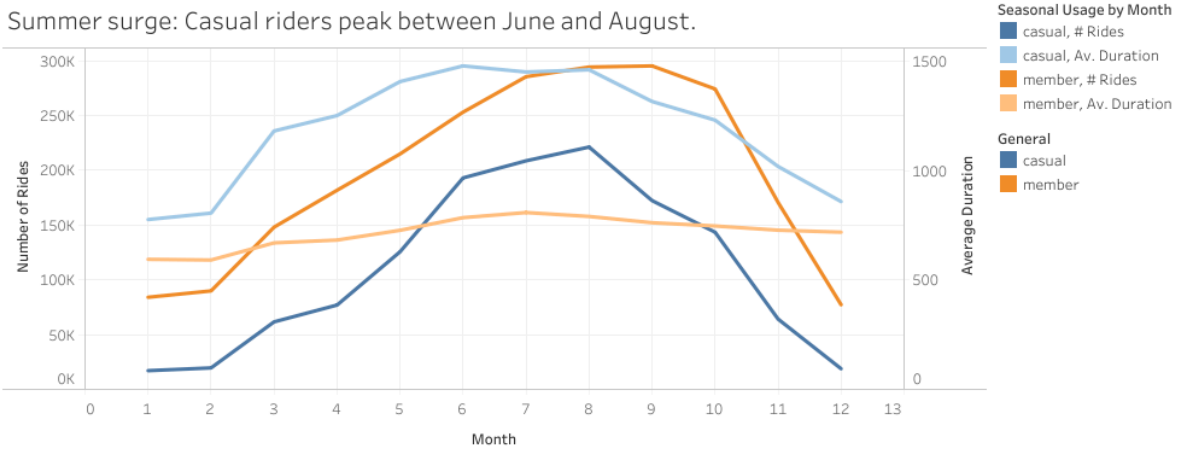
The findings provide a clear answer to “How do annual members and casual riders use Cyclistic bikes differently?”:

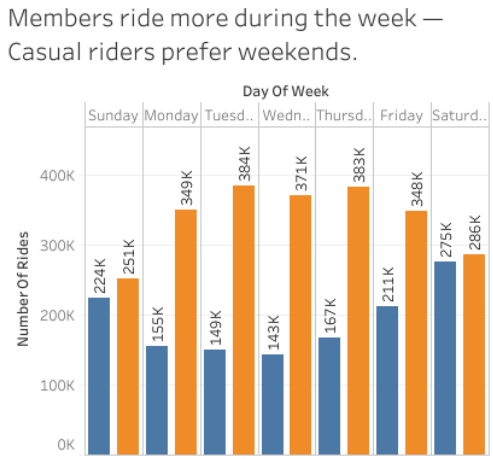
- Members ride frequently, for short durations, mainly on weekdays
  → indicates commuting and routine transportation.

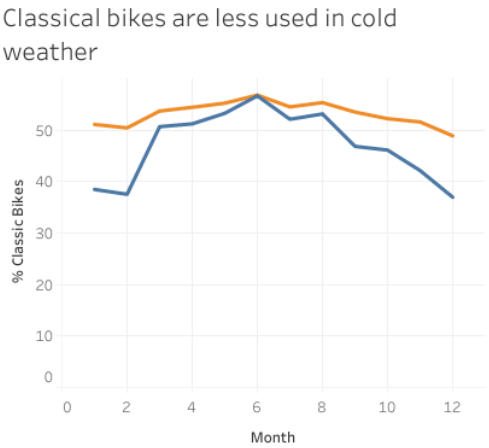
- Casual riders use the service less frequently, but for longer, more recreational rides → primarily weekend and leisure-driven usage.
- Casual riders travel more during seasonal peaks → they may be motivated by tourism, exercise, or exploration.
- Members maintain relatively consistent ridership across months → aligns with stable daily mobility needs.

5. Supporting visualizations and key findings

We elaborated visualizations using Tableau (link). The visualizations clearly show distinct behavioral patterns between annual members and casual riders. By analyzing usage over the week, the year, and by bike type, the data reveals strong and consistent differences that directly address the business question.







The story that emerges from the visualizations is simple and compelling:

- Casual riders behave like leisure users, with long rides, strong weekend activity, and significant peaks in warm months.
- Members behave like commuters, riding mostly during weekdays, taking shorter trips, and showing more stable usage across the year.
- Bike preferences change with temperature, particularly for classic bikes, which drop in usage during colder months, especially for casual riders.

More specifically, the insight we gain are:

1. Seasonal Usage by Month (Top Chart)

Insight: Casual ridership surges from June to August, both in ride counts and duration. Members also increase usage in summer, but much more moderately.
This clearly shows that summer campaigns can target highly engaged casual riders.

2. Weekday vs. Weekend Usage (Bottom Left Chart)

Insight:This visual confirms that members and casuals ride at different times for different purposes, a core behavioral distinction:

- Members: highest ridership Monday–Friday, aligned with commuting.
- Casual riders: highest activity Saturday–Sunday, aligned with leisure.

3. Classic Bike Share by Month (Bottom Right Chart)

Insight: Classic bike usage is highest in warm months (peaking mid-summer) and drops in colder weather. This suggests that winter campaigns could highlight electric bikes, which are less affected by cold conditions.

6. Your top three recommendations based on your analysis

The analysis clearly shows that casual riders and annual members use Cyclistic bikes in fundamentally different ways. Members ride more frequently, mostly on weekdays, for shorter and more consistent durations—indicating commuting or utilitarian trips. Casual riders, however, ride longer, more often on weekends, and display strong seasonality, with their usage peaking sharply during summer months.

These differences reveal specific windows of opportunity for converting casual riders into annual members through targeted marketing strategies.

Recommendation 1 — Launch a Summer Conversion Campaign Targeting High-Engagement Casual Riders

- Why:

Summer months (June–August) show the highest ridership for casual users, both in number of rides and duration. This is when casual riders already demonstrate strong interest in the service.

- What to do:

Offer seasonal promotions such as:

- “Summer Unlimited Rides Pass” with discounted annual membership rollover.
- Limited-time membership discounts for riders taking more than X trips in a month.
- Use in-app notifications, QR codes at high-traffic stations, and email reminders after long rides.

- Expected impact:

Converting casual riders when they are most engaged maximizes the likelihood of long-term retention and increases membership volume during the busiest months.

Recommendation 2 — Promote Membership as the Best Option for Weekday Riders and Short, Frequent Trips

- Why:

Analysis shows that members ride heavily during weekdays and take shorter trips, a pattern consistent with commuting or routine mobility. Casual weekday riders behave similarly, but pay more per use.

- What to do:

Target weekday casual riders with:

- "Commute smarter" membership messaging
- Calculators comparing pay-per-ride vs. member savings
- First-month-free or low-cost trial memberships
- Place digital ads near workplaces, transit hubs, and downtown stations.

- Expected impact:

Positioning the annual membership as a commuter tool encourages habitual weekday casual riders to convert, improving revenue stability and reducing dependence on seasonal peaks.

Recommendation 3 — Use Seasonal Bike Type Trends to Encourage Year-Round Membership

- Why:

The data shows that classic bike usage drops in colder months, while electric bike usage becomes more prominent. Riders may perceive winter riding as more difficult without motor assistance.

- What to do:

Offer winter benefits such as:

- Free or discounted e-bike unlocks for new members
- Winter-riding tips and safety campaigns
- Cold-weather incentives (bonus ride credits)
- Promote the reliability and comfort of electric bikes during the winter season.

- Expected impact:

These incentives help maintain ridership during low-activity months and position membership as a year-round mobility solution, not just a summer activity.

Next Steps

- Conduct A/B testing on messaging
- Analyze origin/destination station trends
- Gather user feedback via surveys
- Integrate external datasets (weather, local events, transit data)

I. Annex 1: Power Query

```
let
```

```
    Origen =
Csv.Document(File.Contents("Path\202501-divvy-tripdata.csv"),[Delimiter=
",", Columns=13, Encoding=1252, QuoteStyle=QuoteStyle.None]),
    #"Encabezados promovidos" = Table.PromoteHeaders(Origen,
[PromoteAllScalars=true]),
    #"Tipo cambiado" = Table.TransformColumnTypes(#"Encabezados
promovidos",{{"ride_id", type text}, {"rideable_type", type text},
{"started_at", type datetime}, {"ended_at", type datetime},
{"start_station_name", type text}, {"start_station_id", type text},
{"end_station_name", type text}, {"end_station_id", type text},
{"start_lat", type number}, {"start_lng", type number}, {"end_lat", type
number}, {"end_lng", type number}, {"member_casual", type text}}),
    #"Duplicados quitados" = Table.Distinct(#"Tipo cambiado",
{"ride_id"}),
    #"Null end_station filter" = Table.SelectRows(#"Duplicados
quitados", each [end_station_name] <> null and [end_station_name] <>
""),
    #"Null start_station filter" = Table.SelectRows(#"Null end_station
filter", each [start_station_name] <> null and [start_station_name] <>
""),
    #"Compare time" = Table.AddColumn(#"Null start_station filter",
"ride_time_length", each [ended_at] - [started_at]),
    #"Time length positive filter" = Table.SelectRows(#"Compare time",
each [ride_time_length] > #duration(0, 0, 0, 0)),
    #"Day of week" = Table.AddColumn(#"Time length positive filter",
"day_of_week", each Date.DayOfWeek(Date.From([started_at])) + 1),
    #"Distance stations column" = Table.AddColumn(#"Day of week",
"distance_stations", each Number.Sqrt(Number.Power(
([end_lat]-[start_lat])*111,2) + Number.Power(([end_lng]-[start_lng])*
111 * Number.Cos(([end_lat] + [start_lat]) / 2 * Number.PI / 180),2) ))
in
    #"Distance stations column"
```

II. Annex 2: Python

```python
import pandas as pd
import time

# ============================================================
# 1. TRANSFORM EXCEL SHEETS TO A SINGLE CSV FILE (OPTIONAL STEP)
# ============================================================

# File paths
file_path = "ciclistic_clean_data_v03.xlsx"
```

```
output_csv = "ciclistic_clean_data.csv"

# Data types for reading the Excel sheets
dtype_map = {
    "ride_id": "string",
    "rideable_type": "category",
    "start_station_name": "string",
    "start_station_id": "string",
    "end_station_name": "string",
    "end_station_id": "string",
    "member_casual": "category",

    "start_lat": "float32",
    "start_lng": "float32",
    "end_lat": "float32",
    "end_lng": "float32",

    "distance_stations": "float32",
    "day_of_week": "int8",
}

# Columns to parse as datetime
date_cols = ["started_at", "ended_at"]

# ------------------------------------------------------------------
# NOTE: This block converts all Excel sheets into a single CSV file.
# It is kept here for reference but does not run by default.
# ------------------------------------------------------------------
"""
xls = pd.ExcelFile(file_path, engine="openpyxl")
first_sheet = True

for sheet in xls.sheet_names:
    print(f"Processing: {sheet}")
    start_time = time.time()

    df = pd.read_excel(
        xls,
        sheet_name=sheet,
        dtype=dtype_map,
        parse_dates=date_cols
    )
```

```
    df["ride_time_length"] = pd.to_timedelta(df["ride_time_length"])

    df.to_csv(
        output_csv,
        mode="w" if first_sheet else "a",
        index=False,
        header=first_sheet
    )

    first_sheet = False
    del df

    print(f"Execution time: {time.time() - start_time:.2f} seconds")
"""

# ============================================================
# 2. LOAD CLEANED DATA USING PARQUET (FAST & EFFICIENT)
# ============================================================

# If needed, convert CSV → Parquet (much faster to load)
"""
df = pd.read_csv(
    "ciclistic_clean_data.csv",
    dtype=dtype_map,
    parse_dates=date_cols
)

df["ride_time_length"] = pd.to_timedelta(df["ride_time_length"])
df.to_parquet("ciclistic_clean_data.parquet", compression="snappy")
"""

# Load the already-created parquet file
start_time = time.time()
all_trips = pd.read_parquet("ciclistic_clean_data.parquet")
print(f"Loaded Parquet file in {time.time() - start_time:.2f} seconds")

# ============================================================
# 3. INITIAL ANALYSIS & ADDITIONAL TRANSFORMATIONS
# ============================================================

# Create month and day columns
```

```
all_trips["month"] = all_trips["started_at"].dt.month
all_trips["day"] = all_trips["started_at"].dt.day

# Recalculate ride_time_length in seconds
all_trips["ride_time_length"] = (
    all_trips["ended_at"] - all_trips["started_at"]
).dt.total_seconds()

# Check for negative durations
print("Any negative ride durations?:", any(all_trips["ride_time_length"] < 0))

# Basic inspection
print("\nColumn names:", all_trips.columns)
print("\nTotal rows:", len(all_trips))
print("\nShape:", all_trips.shape)
print("\nFirst 6 rows:\n", all_trips.head())
print("\nColumn types and memory usage:")
all_trips.info(memory_usage="deep")
print("\nNumeric summary:\n", all_trips.describe())

# =============================================================
# 4. DESCRIPTIVE ANALYSIS
# =============================================================

print("\nValue counts for 'member_casual':\n",
    all_trips["member_casual"].value_counts())

print("\nRide length statistics (seconds):\n",
    all_trips["ride_time_length"].describe())

print("\nDistance statistics (km):\n",
    all_trips["distance_stations"].describe())

# Compare ride lengths by user type
print("\nRide length by membership type:\n",
    all_trips.groupby("member_casual", observed=True)["ride_time_length"]
    .agg(["mean", "median", "max", "min"]))

# Compare distances by user type
print("\nDistance by membership type:\n",
    all_trips.groupby("member_casual", observed=True)["distance_stations"]
    .agg(["mean", "median", "max", "min"]))
```

```
# ============================================================
# 5. ANALYSIS BY DAY OF WEEK
# ============================================================

# Map numbers → day names
num_to_day = {
    1: "Sunday", 2: "Monday", 3: "Tuesday",
    4: "Wednesday", 5: "Thursday",
    6: "Friday", 7: "Saturday"
}

all_trips["day_of_week"] = all_trips["day_of_week"].map(num_to_day)

# Order categories
days_order = ["Sunday", "Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday", "Saturday"]

all_trips["day_of_week"] = pd.Categorical(
    all_trips["day_of_week"],
    categories=days_order,
    ordered=True
)

print("\nAverage ride length by user type and weekday:\n",
    all_trips.groupby(["member_casual", "day_of_week"], observed=True)
    ["ride_time_length"].mean())

# Detailed summary by weekday
weekday_summary = all_trips.groupby(
    ["member_casual", "day_of_week"], observed=True
).agg(
    number_of_rides=("ride_id", "count"),
    number_of_classic_bikes=("rideable_type", lambda x: (x ==
"classic_bike").sum()),
    average_duration=("ride_time_length", "mean"),
    average_distance=("distance_stations", "mean"),
).reset_index()

weekday_summary["pct_classic_bikes"] = (
    weekday_summary["number_of_classic_bikes"]
    / weekday_summary["number_of_rides"]
```

```
    * 100)

print("\nSummary of rides by user type and weekday:\n",
    weekday_summary)

# ============================================================
# 6. ANALYSIS BY MONTH
# ============================================================

monthly_summary = all_trips.groupby(
    ["member_casual", "month"], observed=True
).agg(
    number_of_rides=("ride_id", "count"),
    number_of_classic_bikes=("rideable_type", lambda x: (x ==
"classic_bike").sum()),
    average_duration=("ride_time_length", "mean"),
    average_distance=("distance_stations", "mean")
).reset_index()

monthly_summary["pct_classic_bikes"] = (
    monthly_summary["number_of_classic_bikes"]
    / monthly_summary["number_of_rides"]
    * 100)

print("\nSummary of rides by user type and month:\n",
    monthly_summary)

# ============================================================
# 7. EXPORT SUMMARY FILE
# ============================================================
# Create a .csv file to visualize elsewhere
weekday_summary.to_csv('weekday_summary.csv', index=False)
monthly_summary.to_csv('monthly_summary.csv', index=False)
```